Practical Machine Learning Assignment

Daniela Curl

9/18/2019

Step 1

Load ALL THE LIBRARIES. You know. Just because.

```
library(ElemStatLearn)
## Warning: package 'ElemStatLearn' was built under R version 3.5.2
library(lubridate)
##
## Attaching package: 'lubridate'
## The following object is masked from 'package:base':
##
##     date
library(AppliedPredictiveModeling)
library(ElemStatLearn)
library(randomForest)
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
library(caret)
## Warning: package 'caret' was built under R version 3.5.2
## Loading required package: lattice
## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 3.5.2
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:randomForest':
##
##     margin
library(e1071)
## Warning: package 'e1071' was built under R version 3.5.2
library(forecast)
## Warning: package 'forecast' was built under R version 3.5.2
```

Step 2

Read in Data. Can't do any analysis without any data!

```
'Read in Training Data'
```

```
## [1] "Read in Training Data"
```

```
training_all = read.csv("~/Desktop/pml-training.csv")
```

```
'Read in Testing Data'
```

```
## [1] "Read in Testing Data"
```

```
testing_final = read.csv("~/Desktop/pml-testing.csv")
```

Step 3

Segregate Data! I'm using 70% of my data for training, and 30% to Test.

```
inTrain <- createDataPartition(y=training_all$classe, p=0.7, list=FALSE)


training <- training_all[inTrain, ]


testing <- training_all[-inTrain, ]
```

Step 4

Data Clean-up! Things I tried:

- Removing all rows with NA data via omit.na (not so great, left me with only 406 observations).
- Remove all columns that had mostly NA values, plus the first column which just numbered the rows. I wanted to see if using most of the columns would produce a good model.
- Removed all but a couple of the beginning columns (kept 2.username through 11.total_accel_belt, and 157.magnet_forearm_x through 160. classe). Conversely, I wanted to see if I could produce a good model using only a small fraction of the data.

```
'Remove all rows with NA values'
```

```
## [1] "Remove all rows with NA values"
```

```
training_omitna <- na.omit(training)


'Remove Columns with mostly NA values'
```

```
## [1] "Remove Columns with mostly NA values"
```

```
training_reduced <- training[ -c(1,12:36, 50:59, 69:83, 87:101, 103:112, 125:139, 141:150
) ]


'Remove all columns except for columns 2 through 11 and 157 through 160'
```

```
## [1] "Remove all columns except for columns 2 through 11 and 157 through 160"
```

```
training_really_reduced <- training[ -c(1, 12:156) ]
```

Step 5

Train the models!

- First I trained a random forest model for the training_omitna dataset. (Spoiler: the prediction for my testing dataset returned all As)
- Then I trained a random forest model for the training_reduced dataset. (Spoiler: Jackpot - correct answeres!)
- Finally, I trained another random forest model on the training_mfa dataset. (Spoiler: Double Jackpot - correct answers AND faster!)

NOTE: Due to length of time to re-run these for knitr, I only recompiled my 2nd and 3rd models. The first one takes like 2 hours to train, but since the first one produced all As I'm only retraining the second two for the assignment.

```
set.seed(09222019)
```

```
'modFit1 <- train(classe ~., method = "rf", data = training_omitna)'
```

```
## [1] "modFit1 <- train(classe ~., method = \"rf\", data = training_omitna)"
```

```
modFit2 <- train(classe ~., method = "rf", data = training_reduced)
```

```
modFit3 <- train(classe ~., method = "rf", data = training_really_reduced)
```

Step 6

Prediction time! I spoiled up above but I'll write it out here too:

1. Model 1 using Random Forest and training_omitna - All As
2. Model 2 using Random Forest and training_reduced - B A B A A E D B A A B C B A E E A B B B
3. Model 3 using Random Forest and training_mfa: B A B A A E D B A A B C B A E E A B B B

```
'predict1 <- predict(modFit1, testing)'
```

```
## [1] "predict1 <- predict(modFit1, testing)"
```

```
predict2 <- predict(modFit2, testing_final)
```

```
predict3 <- predict(modFit3, testing_final)
```

```
'predict1'
```

```
## [1] "predict1"
```

```
print(predict2)
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

```
print(predict3)
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
```

```
## Levels: A B C D E
```

Step 7

Accuracies and Reasoning

After comparing the accuracies of both models, I've decided that the model utlizing only 14 columns is (marginally) the better one. It has a better accuracy (99.98% vs 99.96%), and it was faster during the learning process, which is also a benefit.

```
predict4 <- predict(modFit2, testing)
predict5 <- predict(modFit3, testing)


'Confusion Matrix'
## [1] "Confusion Matrix"
confusionMatrix(predict4, testing$classe)$overall[1]
##  Accuracy
## 0.9998301
confusionMatrix(predict5, testing$classe)$overall[1]
##  Accuracy
## 0.9998301
```